What is claimed is:

1. A method for ensuring type safe linkage, comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type;

imposing a constraint on program instructions that the specified type when loaded by a loader that defines the first class is the same as the specified type when loaded by a loader that defines the second class; and

verifying compliance with the constraint.

2. The method of claim 1, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references a field that (iii) is contained in a second class and (iv) is of a specified type.

3. The method of claim 1, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references a method that (iii) is contained in a second class and (iv) is of a specified type.

4. The method of claim 1, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references an overridden method that (iii) is contained in a second class and (ii) is of a specified type.

5. The method of claim 1, wherein verifying compliance with the constraint further comprises verifying compliance with the constraint when the specified type has been loaded by at least one of the loaders that define the first and second classes.

6. A method for ensuring class type safe linkage, comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

imposing a constraint on program instructions that the third class when loaded by a loader that defines the first class is the same as the third class when loaded by a loader that defines the second class; and

verifying compliance with the constraint when the third class has been loaded by at least one of the loaders that define the first and second classes.

7. A method for ensuring class type safe linkage in a runtime environment having a first class that is defined by a first loader, a second class that is defined by a second loader that may be different from the first loader, the first class referencing an attribute that is contained in the second class and that has a type defined by the third class, comprising:

imposing a constraint on program instructions that the third class as loaded by the first loader is the same as the third class as loaded by the second loader; and

verifying compliance with the constraint when the third class has been loaded by at least one of the first loader and the second loader.

8. A method for ensuring class type safe linkage, comprising:

identifying a class <C, L1> that references a field y, of type named E, declared in class <D, L2>;

imposing a constraint that $E^{L1} = E^{L2}$; and

verifying compliance with each of the constraint when at least one of E^{L1} and E^{L2} is loaded during the course of normal execution.

9. A method for ensuring class type safe linkage, comprising:

identifying a class <C, L1> that references a method g() of a class <D, L2>, wherein the method g() accepts as arguments attributes of types named E1, E2, . . . En and has a return type named E0;

imposing constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and

verifying compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

10. A method for ensuring class type safe linkage, comprising:

identifying a class <C, L1> that declares a method g() inherited from a class <D, L2>, wherein the method g() declares argument types named E1, E2, . . . En and has a return type named E0;

imposing constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and verifying compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

11. A method for ensuring type safe linkage, comprising:

identifying a first module that references an attribute that (i) is contained in a second module and (ii) is of a specified type;

imposing a constraint on program instructions that the specified type when loaded by a loader that defines the first module is the same as the specified type when loaded by a loader that defines the second module; and

verifying compliance with the constraint subsequent to imposing the constraint.

12. A method for ensuring class type safe linkage, comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

imposing a constraint on program instructions that a defining loader used by the first class to load the third class be the same defining loader used by the second class to load the third class; and

verifying compliance with the constraint when the third class has been loaded by at least one of the defining loaders of the first and second classes.

13. A method for ensuring class type safe linkage, comprising:

identifying a class <C, L1> that references a field y, of type E, declared in class <D, L2>;

imposing a constraint that $E^{L1} = E^{L2}$; and

verifying compliance with each of the constraints at the earliest possible time without performing any class loading other than that required for normal program execution.

14. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type;

impose a constraint on program instructions that the specified type when loaded by a loader that defines the first class is the same as the specified type when loaded by a loader that defines the second class; and

verify compliance with the constraint.

15. The apparatus of claim 14, wherein the program instructions to identify a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprise program instructions to identify a first class that references a field that (iii) is contained in a second class and (iv) is of a specified type.

16. The apparatus of claim 14, the program instructions to identify a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprise program instructions to identify a first class that references a method that (iii) is contained in a second class and (iv) is of a specified type.

17. The apparatus of claim 14, wherein the program instructions to identify a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprise program instructions to identify a first class that references an overridden method that (iii) is contained in a second class and (ii) is of a specified type.

18. The apparatus of claim 14, wherein the program instructions to verify compliance with the constraint further comprise program instructions to verify compliance with the constraint when the specified type has been loaded by at least one of the loaders that define the first and second classes.

19. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

impose a constraint on program instructions that the third class when loaded by a loader that defines the first class is the same as the third class when loaded by a loader that defines the second class; and

verify compliance with the constraint when the third class has been loaded by at least one of the loaders that define the first and second classes.

20. An apparatus for ensuring class type safe linkage in a runtime environment having a first class that is defined by a first loader, a second class that is defined by a second loader that may be different from the first loader, the first class referencing an attribute that is contained in the second class and that has a type defined by the third class, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

impose a constraint on program instructions that the third class as loaded by the first loader is the same as the third class as loaded by the second loader; and

verify compliance with the constraint when the third class has been loaded by at least one of the first loader and the second loader.

21. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a class <C, L1> that references a field y, of type named E, declared in class <D, L2>;

impose a constraint that $E^{L1} = E^{L2}$; and

verify compliance with each of the constraint when at least one of E^{L1} and E^{L2} is loaded during the course of normal execution.

22. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a class <C, L1> that references a method g() of a class <D, L2>, wherein the method g() accepts as arguments attributes of types named E1, E2, . . . En and has a return type named E0;

impose constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and

verify compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

23. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a class <C, L1> that declares a method g() inherited from a class <D, L2>, wherein the method g() declares argument types named E1, E2, . . . En and has a return type named E0;

impose constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and

verify compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

24. An apparatus for ensuring type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a first module that references an attribute that (i) is contained in a second module and (ii) is of a specified type;

impose a constraint on program instructions that the specified type when loaded by a loader that defines the first module is the same as the specified type when loaded by a loader that defines the second module; and

verify compliance with the constraint subsequent to imposing the constraint.

25. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

impose a constraint on program instructions that a defining loader used by the first class to load the third class be the same defining loader used by the second class to load the third class; and

verify compliance with the constraint when the third class has been loaded by at least one of the defining loaders of the first and second classes.

26. An apparatus for ensuring class type safe linkage, comprising:

at least one memory having program instructions, and

at least one processor configured to use the program instructions to:

identify a class <C, L1> that references a field y, of type E, declared in class <D, L2>;

impose a constraint that $E^{L1} = E^{L2}$; and

verify compliance with each of the constraints at the earliest possible time without performing any class loading other than that required for normal program execution.

27. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type;

imposing a constraint on program instructions that the specified type when loaded by a loader that defines the first class is the same as the specified type when loaded by a loader that defines the second class; and

verifying compliance with the constraint.

28. The computer readable medium of claim 27, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references a field that (iii) is contained in a second class and (iv) is of a specified type.

29. The method of claim 27, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references a method that (iii) is contained in a second class and (iv) is of a specified type.

30. The method of claim 27, wherein identifying a first class that references an attribute that (i) is contained in a second class and (ii) is of a specified type further comprises identifying a first class that references an overridden method that (iii) is contained in a second class and (ii) is of a specified type.

31. The method of claim 27, wherein verifying compliance with the constraint further comprises verifying compliance with the constraint when the specified type has been loaded by at least one of the loaders that define the first and second classes.

32. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

imposing a constraint on program instructions that the third class when loaded by a loader that defines the first class is the same as the third class when loaded by a loader that defines the second class; and

verifying compliance with the constraint when the third class has been loaded by at least one of the loaders that define the first and second classes.

33. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage in a runtime environment having a first class that is defined by a first loader, a second class that is defined by a second loader that may be different from the first loader, the first class referencing an attribute that is contained in the second class and that has a type defined by the third class, the method comprising:

imposing a constraint on program instructions that the third class as loaded by the first loader is the same as the third class as loaded by the second loader; and

verifying compliance with the constraint when the third class has been loaded by at least one of the first loader and the second loader.

34. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a class <C, L1> that references a field y, of type named E, declared in class <D, L2>;

imposing a constraint that $E^{L1} = E^{L2}$; and

verifying compliance with each of the constraint when at least one of E^{L1} and E^{L2} is loaded during the course of normal execution.

35. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a class <C, L1> that references a method g() of a class <D, L2>, wherein the method g() accepts as arguments attributes of types named E1, E2, . . . En and has a return type named E0;

imposing constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and

verifying compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

36. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a class <C, L1> that declares a method g() inherited from a class <D, L2>, wherein the method g() declares argument types named E1, E2, . . . En and has a return type named E0;

imposing constraints that $E0^{L1} = E0^{L2}$, $E1^{L1} = E1^{L2}$, . . . $En^{L1} = En^{L2}$; and

verifying compliance with each of the constraints when at least one of En^{L1} and En^{L2} has both been loaded.

37. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a first module that references an attribute that (i) is contained in a second module and (ii) is of a specified type;

imposing a constraint on program instructions that the specified type when loaded by a loader that defines the first module is the same as the specified type when loaded by a loader that defines the second module; and

verifying compliance with the constraint subsequent to imposing the constraint.

38. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a first class that references an attribute that (i) is contained in a second class and (ii) has a type defined by a third class;

imposing a constraint on program instructions that a defining loader used by the first class to load the third class be the same defining loader used by the second class to load the third class; and

verifying compliance with the constraint when the third class has been loaded by at least one of the defining loaders of the first and second classes.

39. A computer-readable medium containing instructions for controlling a computer system to perform a method for ensuring class type safe linkage, the method comprising:

identifying a class <C, L1> that references a field y, of type E, declared in class <D, L2>;

imposing a constraint that $E^{L1} = E^{L2}$; and

verifying compliance with each of the constraints at the earliest possible time without performing any class loading other than that required for normal program execution.